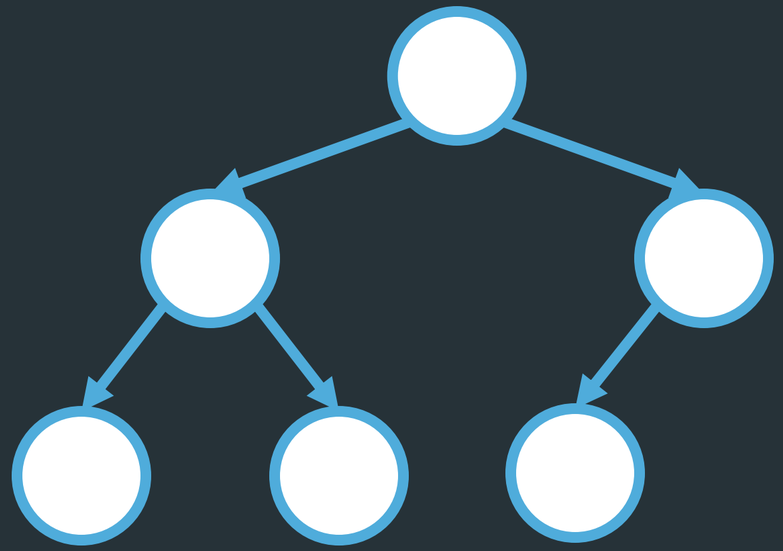
알튜비튜

트리

비선형 자료구조 중 하나인 트리입니다.
그래프의 부분집합으로 계층 관계를 나타낼 때 주로 사용해요.

복습



Tree

- 비선형 자료구조
- 그래프의 부분집합으로 사이클이 없고, V개의 정점에 대해 V-1개의 간선이 있음
- 부모-자식의 계층 구조
- 트리 탐색의 시간 복잡도는 O(h) (h = 트리의 높이)
- 이진 트리와 일반 트리로 나눌 수 있고, 이진 트리는 전위 & 중위 & 후위 순회 가능
- 그래프와 마찬가지로 DFS, BFS를 이용하여 탐색

도전 문제 1

1967번 : 트리의 지름 - Gold 4

문제

- 트리에 존재하는 모든 경로들 중에서 가장 긴 것의 길이를 출력

제한 사항

- N은 1이상 10,000 이하 정수
- 루트 노드의 번호는 항상 1
- 간선의 가중치는 100보다 크지 않은 양의 정수

예제

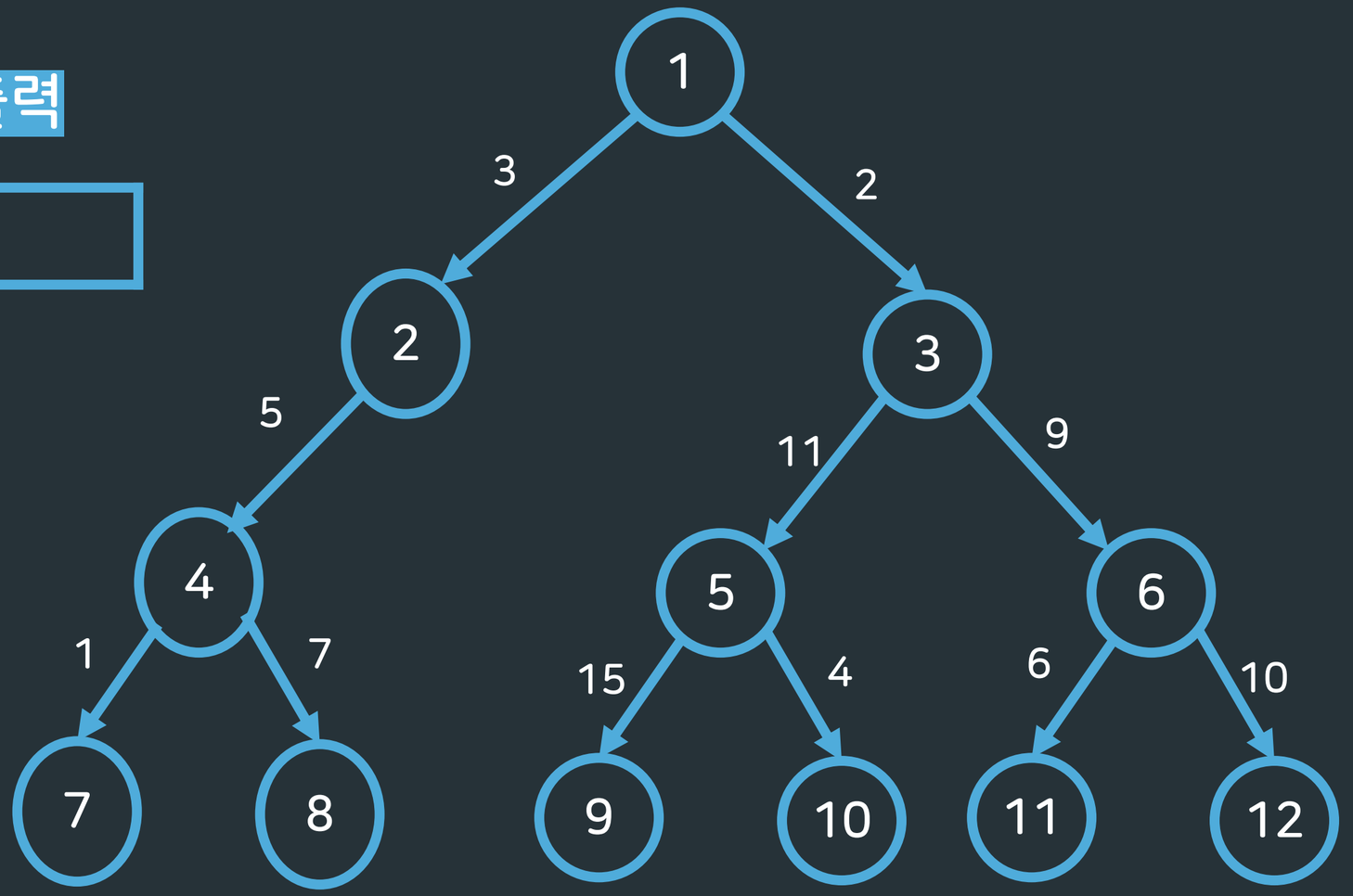
#알튜비튜

예제 입력

```
12
1 2 3
1 3 2
2 4 5
3 5 11
3 6 9
4 7 1
4 8 7
5 9 15
5 10 4
6 11 6
6 12 10
```

예제 출력

```
45
```



몰래 보세요

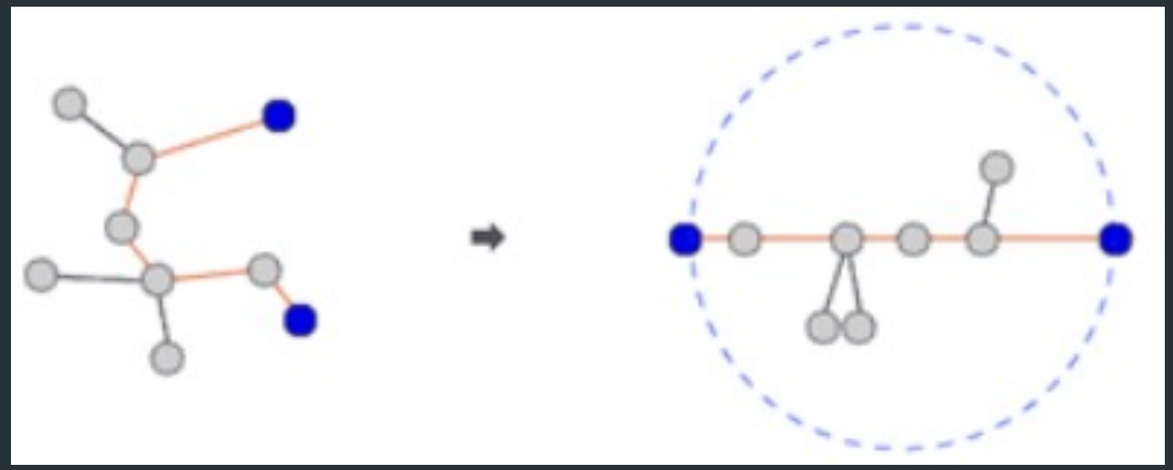
#알튜비튜

Hint

1. 지름을 이루는 두 노드를 기준으로 원을 그리면 임의의 노드들은 모두 원 내부에 존재한다

   ⇒ 임의의 한 노드에서 가장 먼 노드를 찾으면 지름을 이루는 두 노드 중 하나이다

   ⇒ 지름을 이루는 노드에서 다시 가장 먼 노드를 찾으면..?



문제 풀이

#알튜비튜

구현 Point!

- 루트에서 가장 먼 노드를 찾고, 해당 노드에서 다시 가장 먼 노드를 찾자!
- 임의의 노드에서 가장 먼 노드의 특징을 생각해보자!

  ⇒ 만약 노드에 자식이 있다면 자식 노드를 선택하는 것이 더 경로가 길다

  ⇒ 따라서 임의의 노드에서 가장 먼 노드는 자식이 없는 리프 노드이다
- 노드를 탐색할 때 어떤 정보가 필요한가?

  ⇒ 부모 노드의 정보 && 부모 노드까지 누적된 거리
- DFS와 BFS 중 뭘 써야 유리할까?

  ⇒ 부모 노드와 누적된 거리를 따로 저장하지 않기 위해 DFS를 사용하자

도전 문제

24545번 : Y - Platinum 5

문제

- Y-트리의 조건
  1. 4개 이상의 정점과 인접한 정점은 없다.
  2. 인접한 정점의 개수가 3개인 정점은 정확히 하나만 존재한다.
  3. 인접한 정점이 하나뿐인 정점은 정확히 세 개 존재한다.
- Y-트리의 크기는 해당 Y-트리를 이루는 정점의 개수이다.
- 트리가 주어질 때, 정점을 0개 이상 삭제하여 만들 수 있는 가장 큰 Y-트리의 크기를 구하시오.

제한 사항

- 정점의 개수 n은 2 <= n <= 100,000

예제

#알튜비튜

예제 입력 1

```
8
1 2
1 3
2 4
1 5
2 6
2 7
8 7
```

예제 출력 1

```
6
```

예제 입력 2

```
3
1 2
2 3
```

예제 출력 2

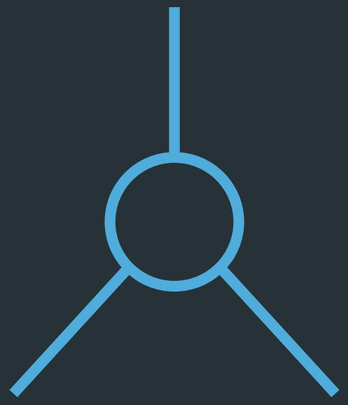
```
0
```

Y-트리는 어떤 모양일까요?

#알튜비튜

1. 4개 이상의 정점과 인접한 정점은 없다.
   → 한 정점의 간선은 최대 3개!
2. 인접한 정점의 개수가 3개인 정점은 정확히 하나만 존재한다.
3. 인접한 정점이 하나뿐인 정점은 정확히 세 개 존재한다.

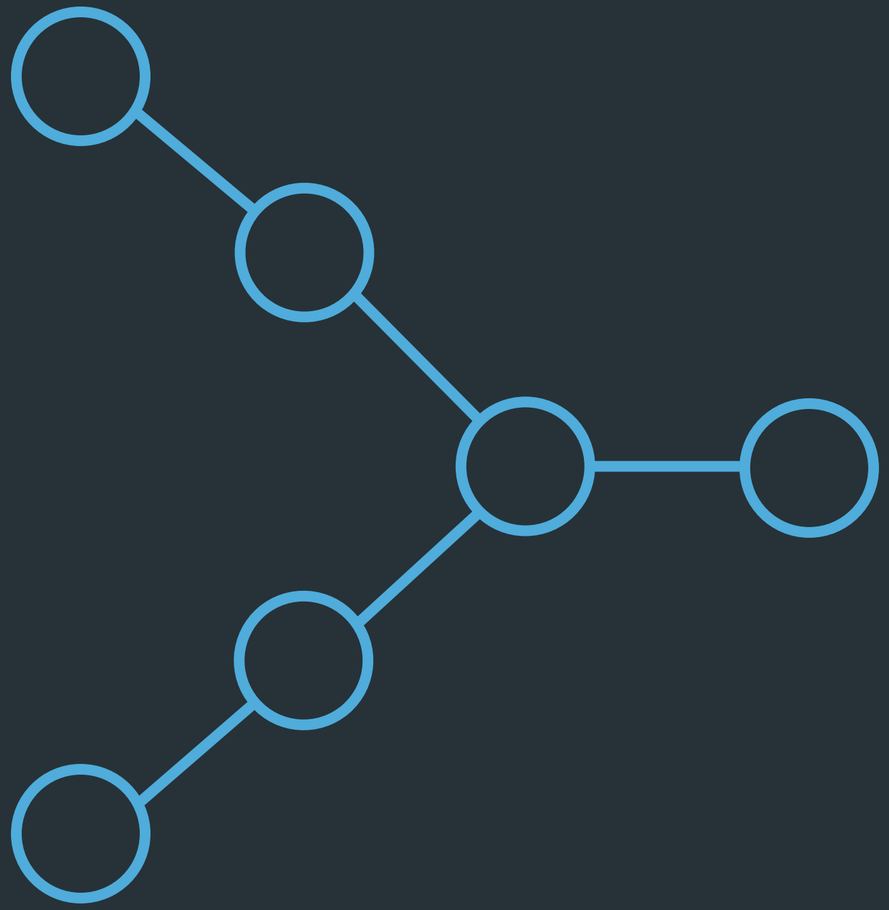
사용할 수 있는 정점의 종류



1개

3개

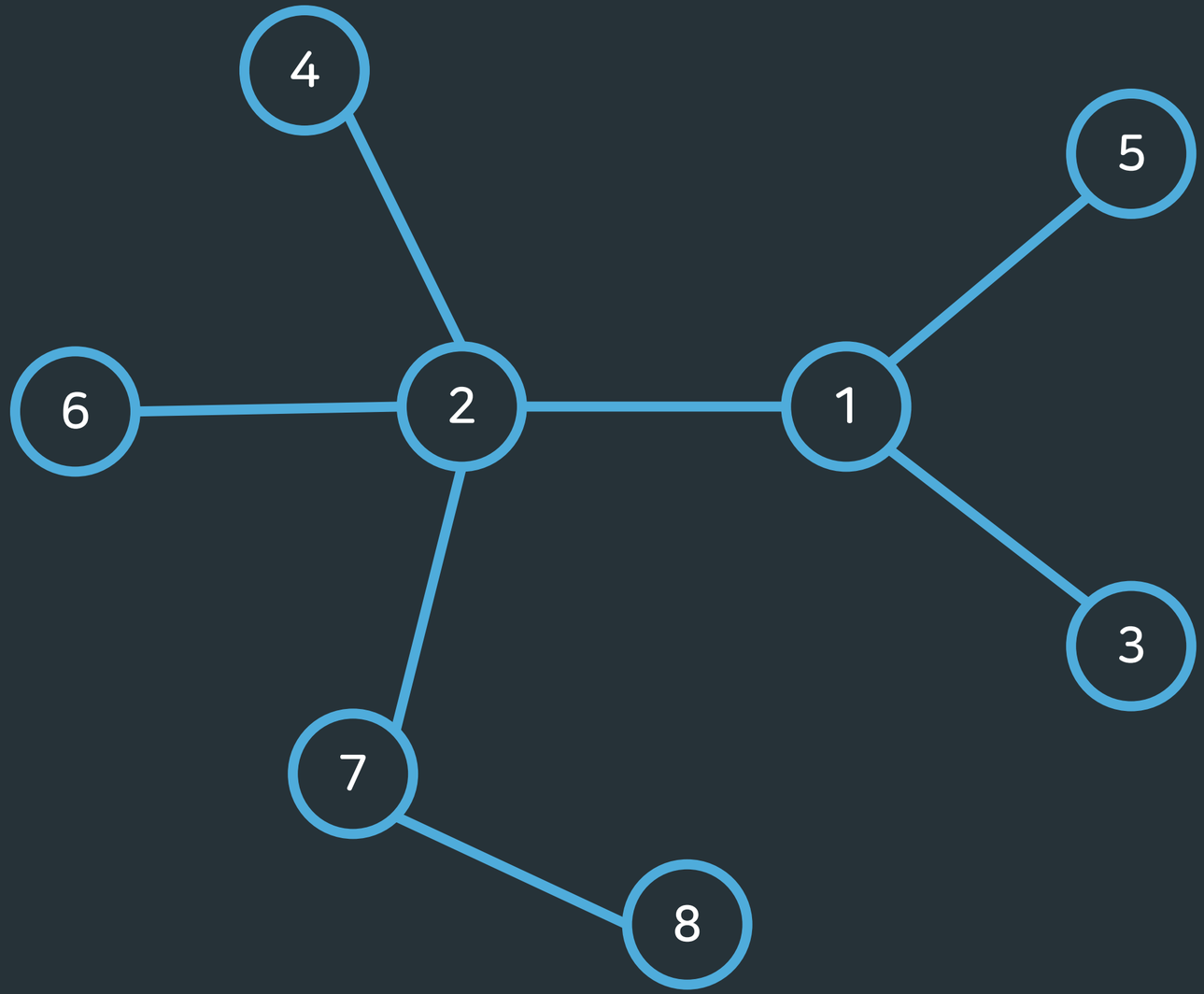
Y-트리는 어떤 모양일까요?



→ Y 모양!

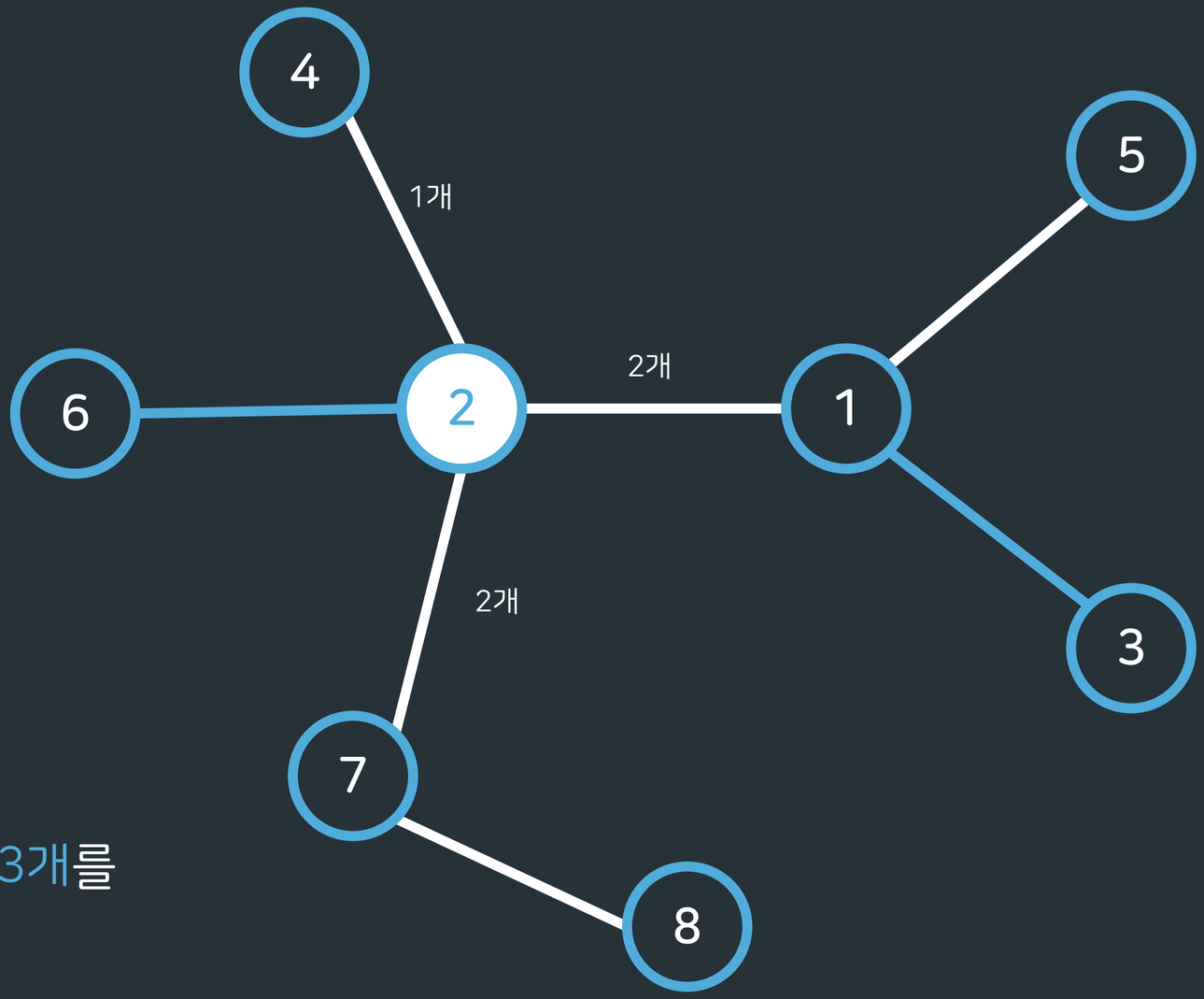
어떤 노드를 삭제하지?

→ 어떤 노드를 선택하지?



어떤 노드를 삭제하지?

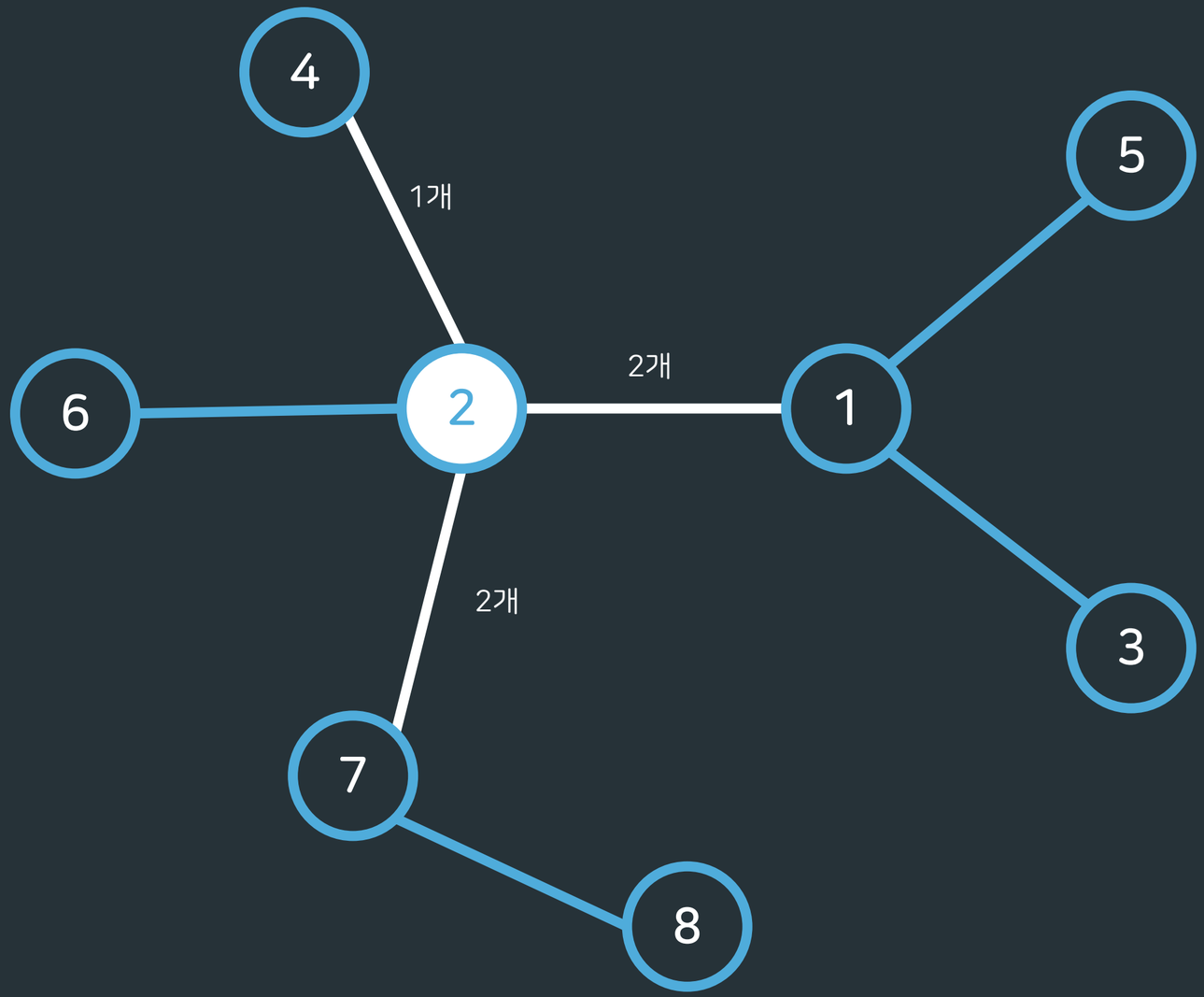
→ 어떤 노드를 선택하지?



중심이 되는 노드 1개 +
해당 노드에 연결된 간선 3개를
고르면 문제 해결

어떤 노드를 삭제하지?

→ 어떤 노드를 선택하지?



→ 한 방향으로 연결된 노드 개수가 가장 많은 간선 3개를 가진 노드를 찾아라

노드의 수가 많은 간선…?

#알튜비튜

노드의 수를 어떻게 세지?

- 앞서 살펴보았던 DFS, BFS로 구현이 가능하지만, 이 문제의 경우 경로에 따라 거쳐가는 노드의 수가 달라짐

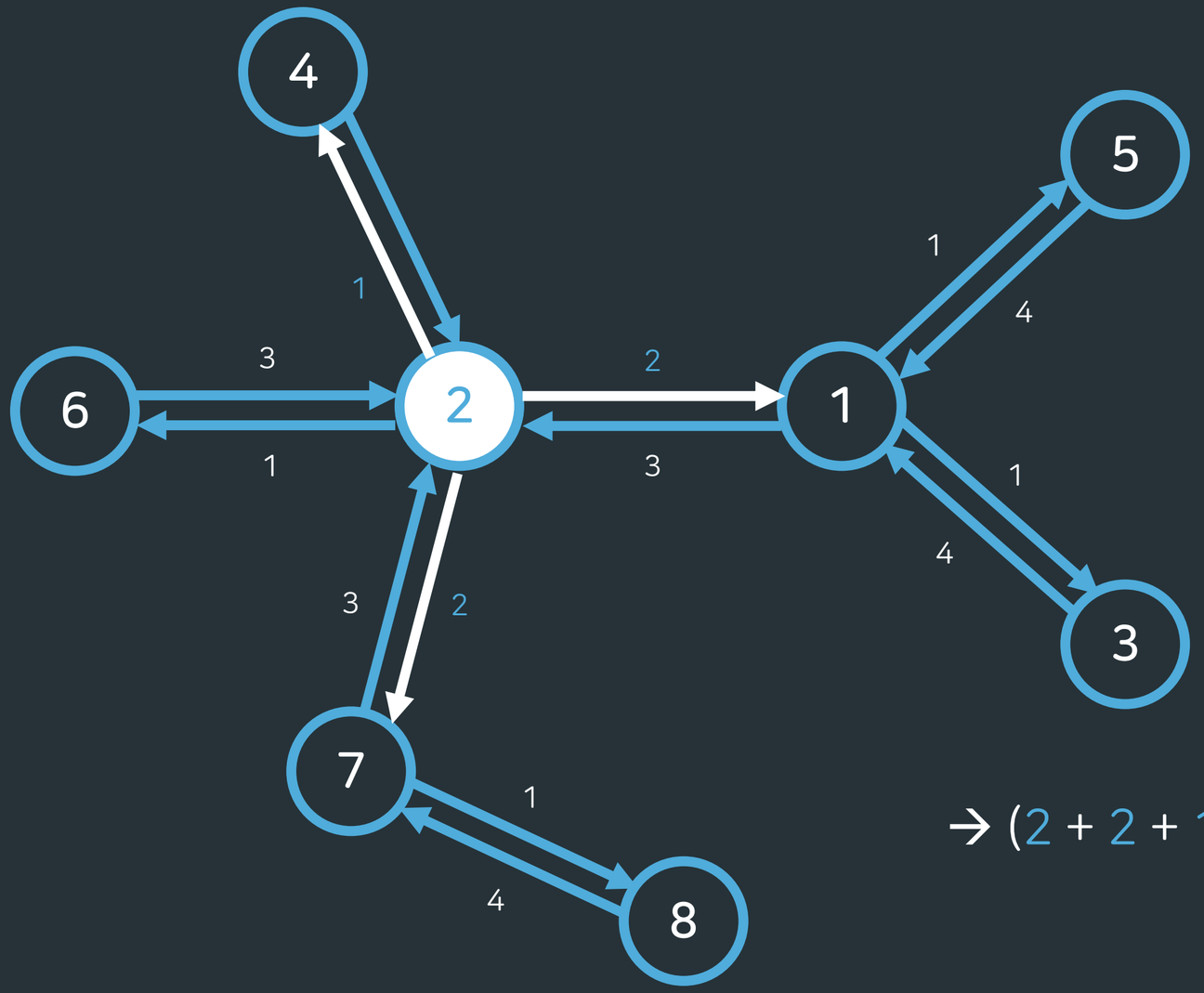
→ 그러나 매번 모든 경우의 수를 탐색하면 시간초과

해결방법

- 메모이제이션 기법을 통해 중복 탐색을 방지
- 트리에서 임의의 두 점을 선택할 경우 경로는 오직 하나 뿐이다.

→ 어떠한 간선을 통해 특정 리프 노드로 이동할 경우, 그 경로와 그 사이의 정점의 수는 고정
→ 간선에 노드의 수를 기록

메모이제이션



→ (2 + 2 + 1) + 1 = 6

간선을 타고 해당 방향으로 이동했을 때 만날 수 있는 노드의 수의 최댓값을 기록

3190번: 뱀 – Gold 4

문제

- 먼저 뱀은 몸길이를 늘려 머리를 다음 칸에 위치시킨다.
- 이동한 칸에 사과가 있다면, 그 칸에 있던 사과가 없어지고 꼬리는 움직이지 않는다.
- 이동한 칸에 사과가 없다면, 몸길이를 줄여서 꼬리가 위치한 칸을 비워준다.
- 뱀이 매 초 주어진 이동방향에 따라 이동할 때, 게임이 끝나는 시간을 구하는 문제

제한 사항

- 보드의 크기 n: $2 \le n \le 100$
- 사과의 개수 k: $0 \le k \le 100$
- 뱀의 방향 변환 정보 개수 l: $1 \le l \le 100$

구현 문제

#알튜비튜

예제 입력 1

```
6
3
3 4
2 5
5 3
3
3 D
15 L
17 D
```

예제 출력 1

```
9
```

예제 입력 2

```
10
4
1 2
1 3
1 4
1 5
4
8 D
10 D
11 D
13 L
```

예제 출력 2

```
21
```

뱀의 머리와 꼬리 위치가 변하네요!

1. 뱀의 머리를 이동 방향으로 한 칸 늘리고
2. 이동한 곳에 사과가 있으면 꼬리의 위치는 그대로!
3. 이동한 곳에 사과가 없으면 꼬리가 위치한 칸 비우기

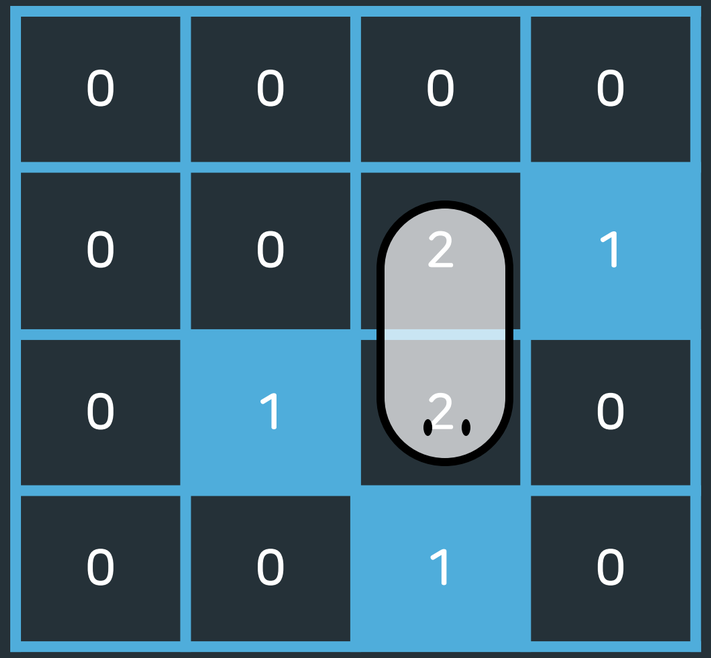
뱀의 머리와 꼬리 위치가 변하네요!

1. 뱀의 머리를 이동 방향으로 한 칸 늘리고
2. 이동한 곳에 사과가 있으면 꼬리의 위치는 그대로!
3. 이동한 곳에 사과가 없으면 꼬리가 위치한 칸 비우기

뱀의 머리와 꼬리 위치가 변하네요!

Deque을 사용해 뱀의 머리부터 꼬리까지 저장해줍시다

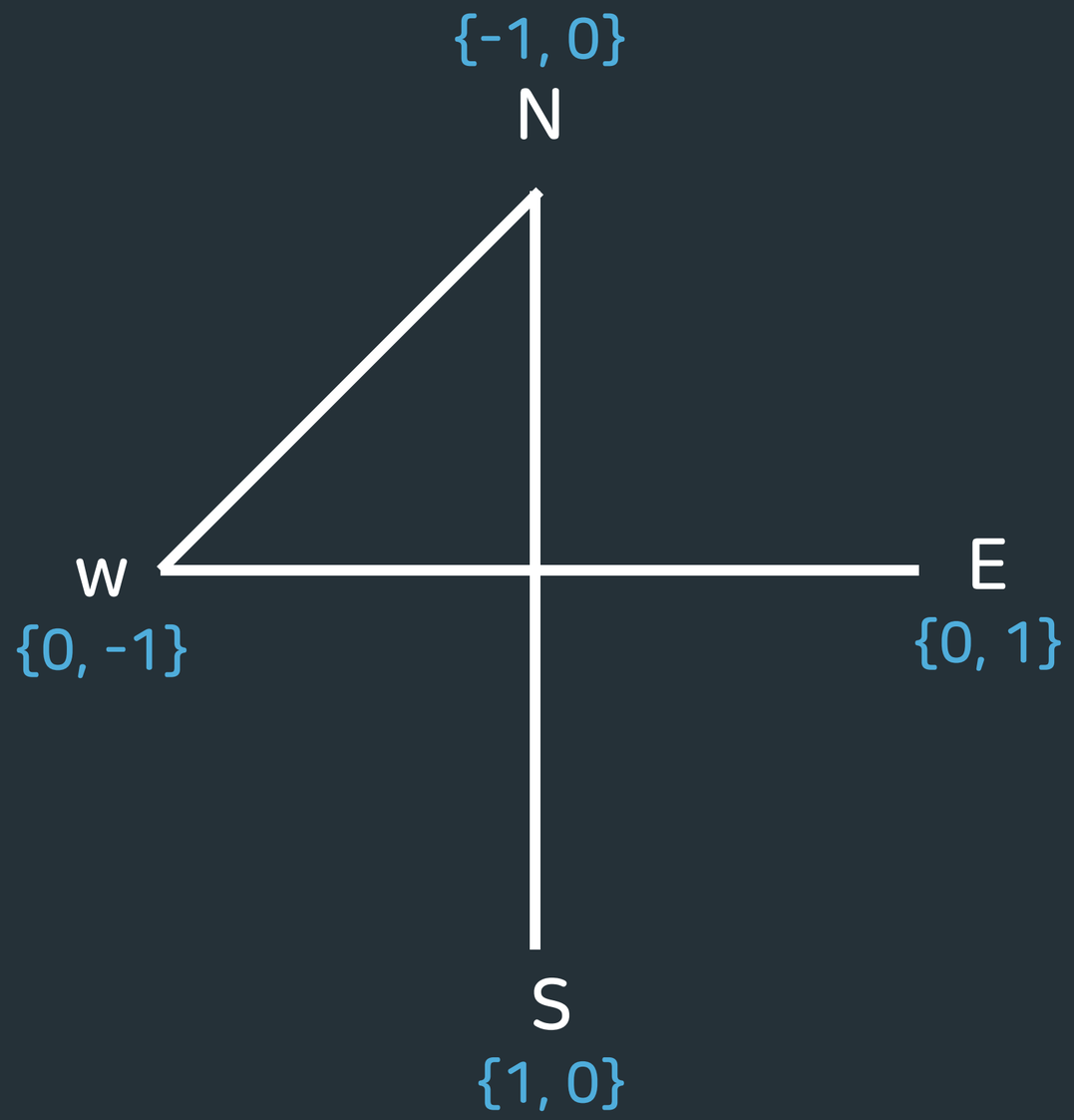
우리끼리 약속



- 아무 것도 없는 칸은 0
- 사과가 있는 칸은 1
- 뱀이 있는 칸은 2

로 표현해줍시다!

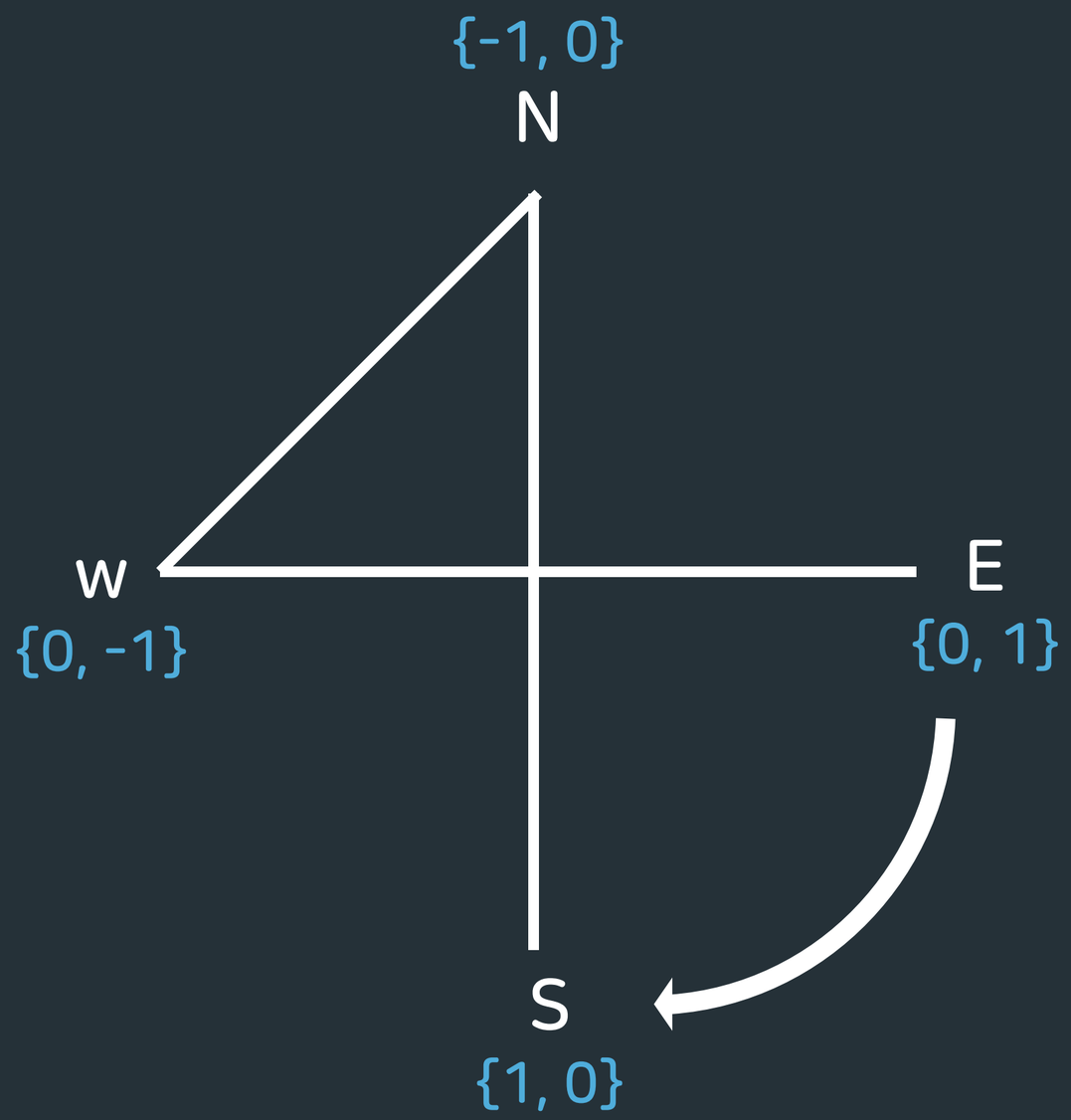
방향 전환



	E	S	W	N
dir	0	1	2	3
dx	0	1	0	-1
dy	1	0	-1	0

- nx = x + dx[dir]
- ny = y + dy[dir]

방향 전환

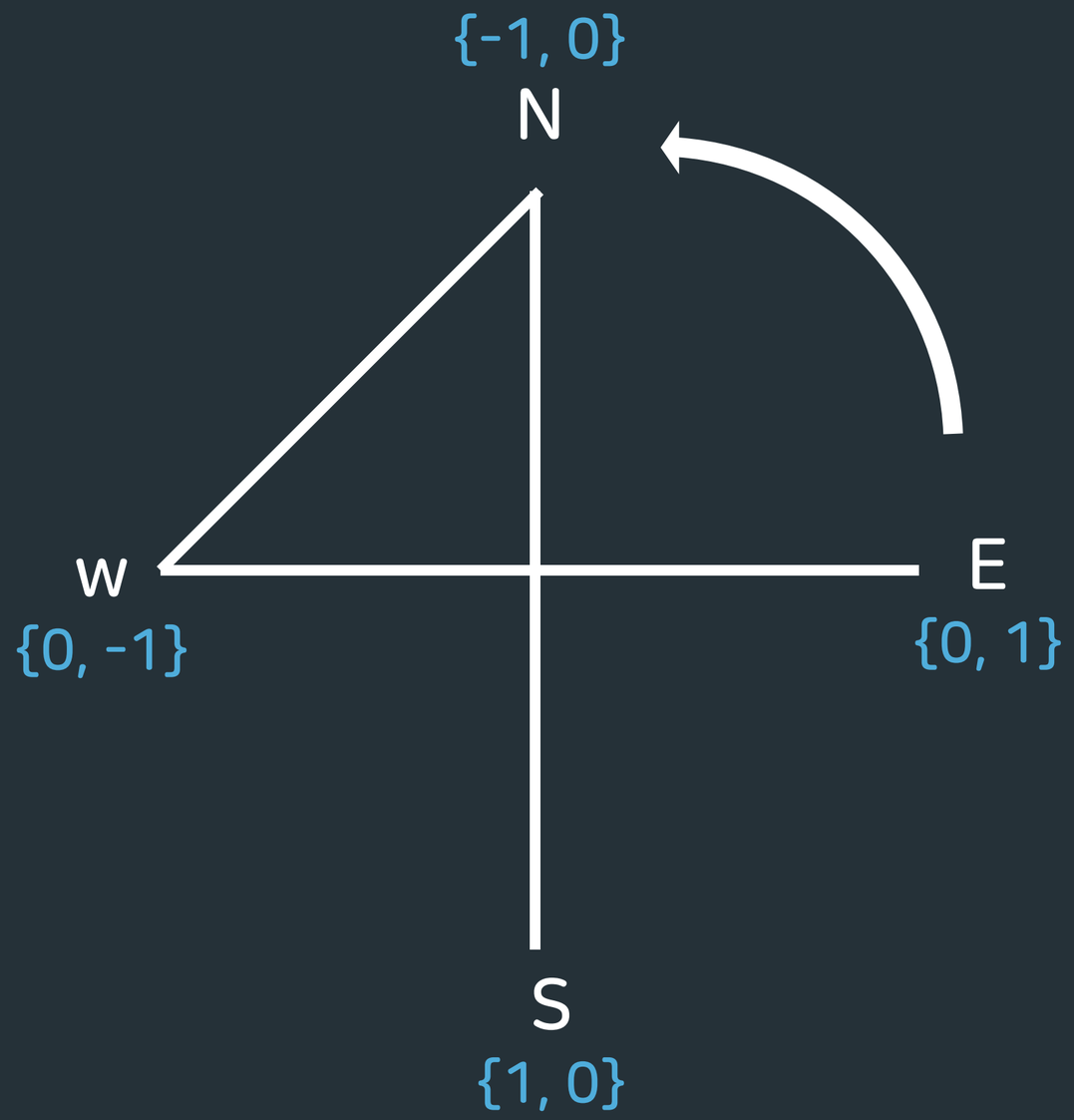


	E	S	W	N
dir	0	1	2	3
dx	0	1	0	-1
dy	1	0	-1	0

- nx = x + dx[dir]
- ny = y + dy[dir]

- dir = (dir + 1) % 4

방향 전환

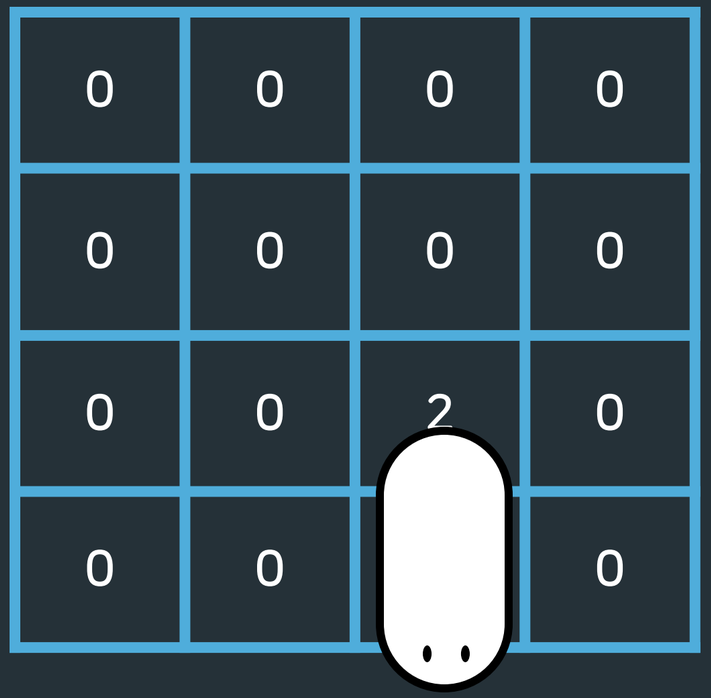


	E	S	W	N
dir	0	1	2	3
dx	0	1	0	-1
dy	1	0	-1	0

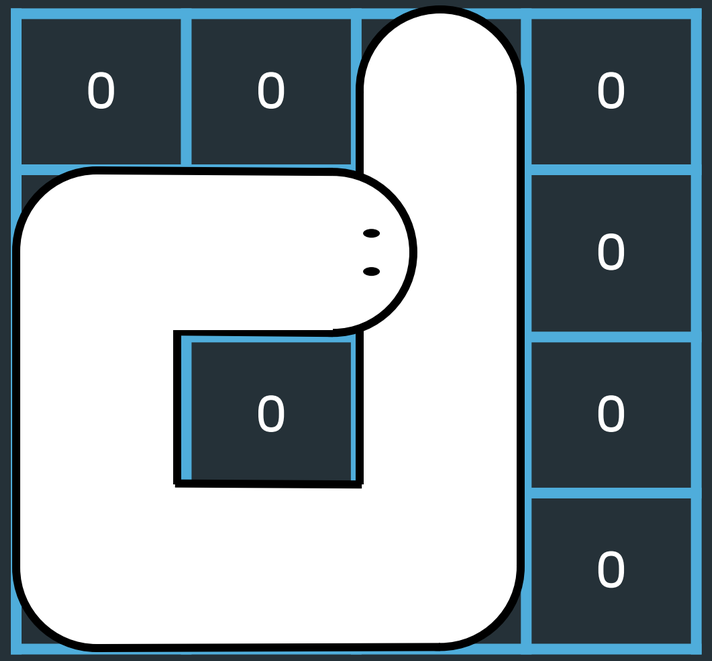
- nx = x + dx[dir]
- ny = y + dy[dir]

- dir = (dir − 1 + 4) % 4
  = (dir + 3) % 4

게임이 종료하는 조건



1.뱀이 보드를 탈출하는 경우
=> 머리의 x, y 좌표가 n보다 큰 경우



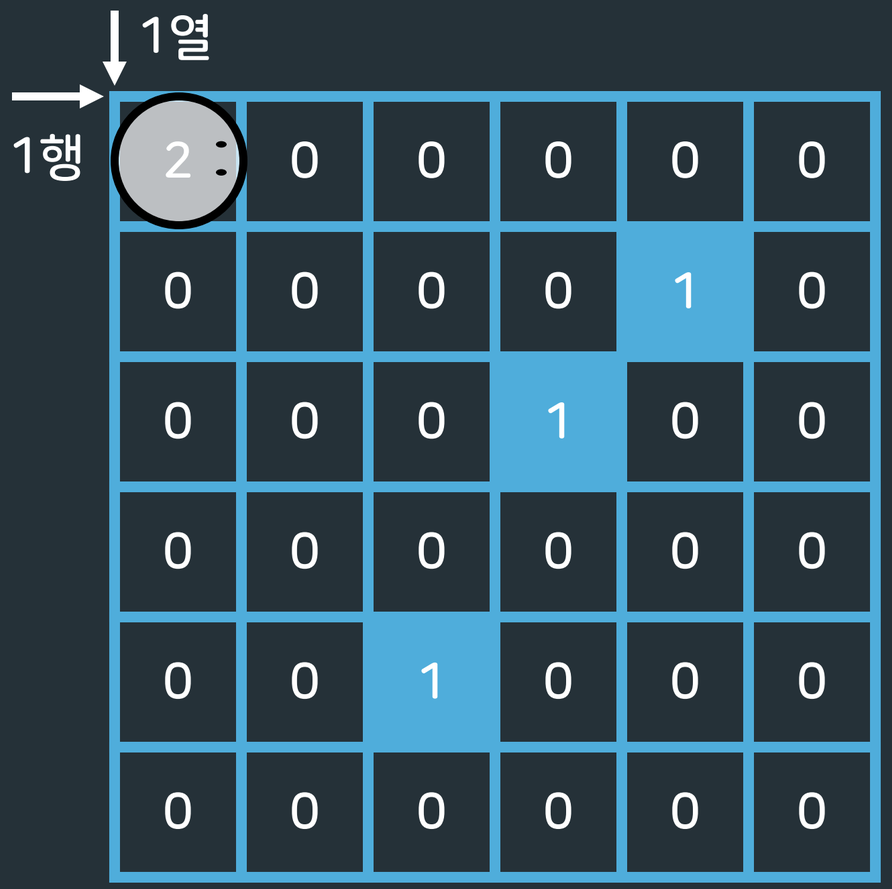
2. 뱀의 머리가 자기 몸에 부딪히는 경우
=> 머리가 이동할 좌표가 이미 2인 경우

차근차근 살펴봅시다

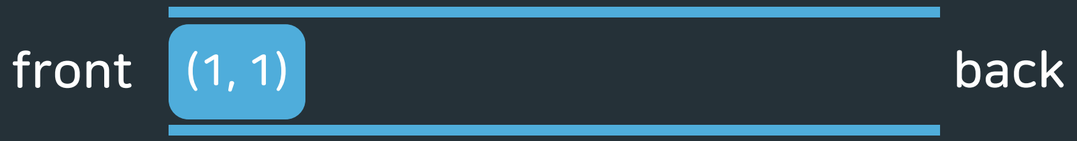
#알튜비튜

방향 변환 정보

3 D
15 L
17 D



- time = 0
- length = 1
- 방향 = 0(동쪽)

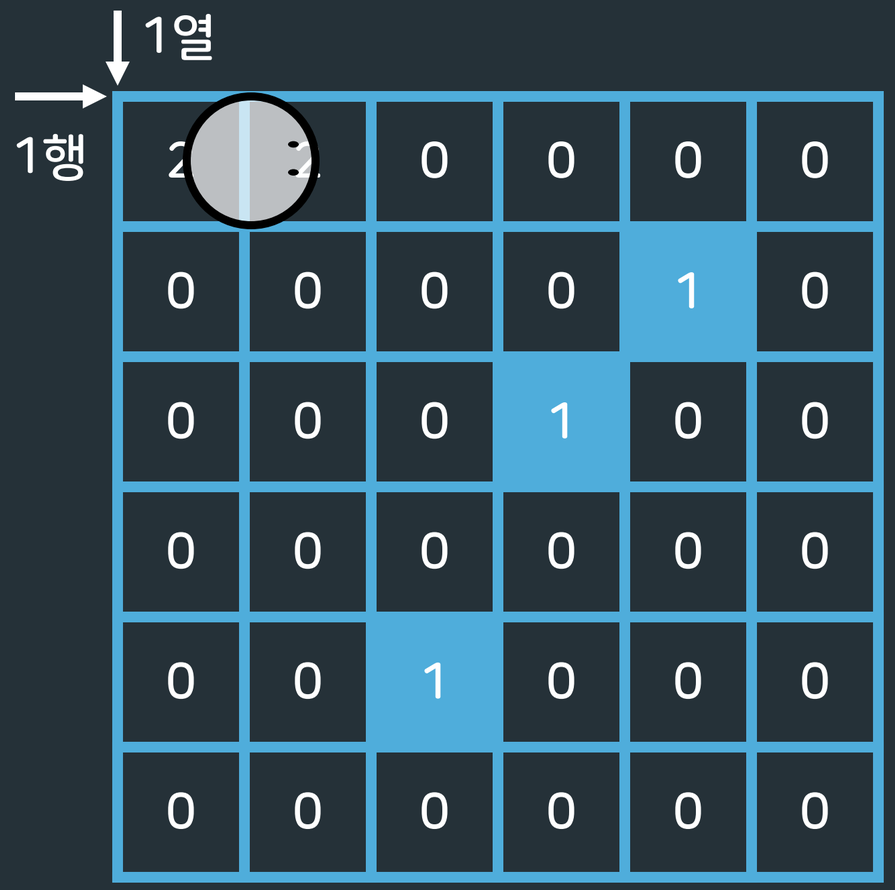


차근차근 살펴봅시다

#알튜비튜

방향 변환 정보

3 D
15 L
17 D



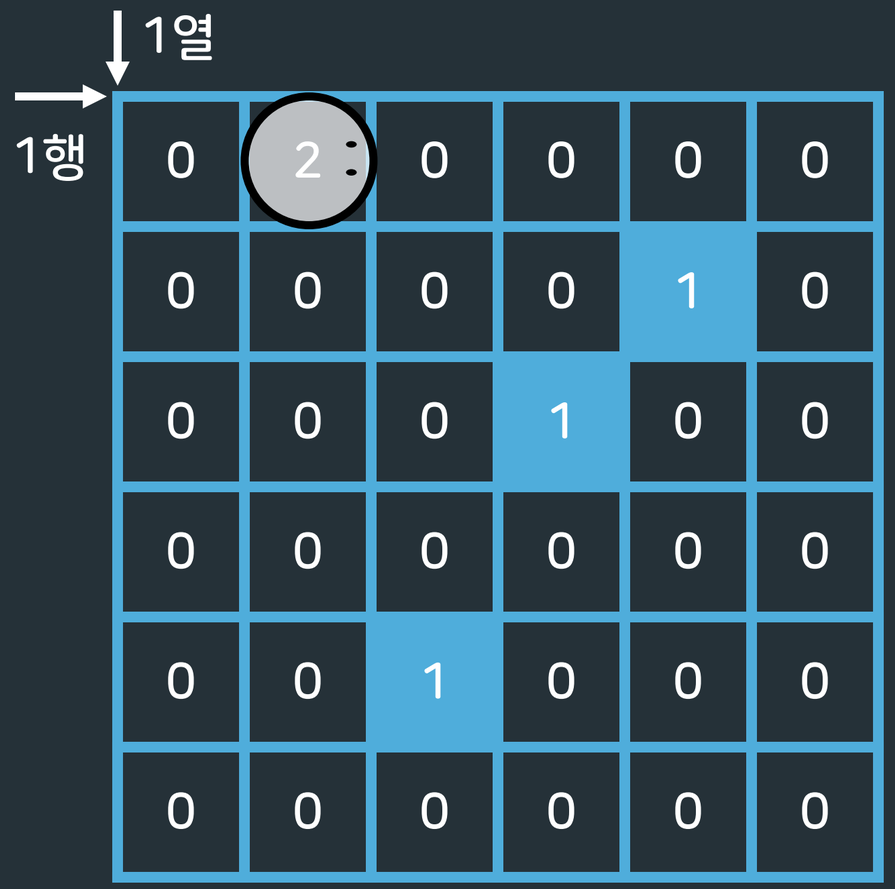
- time = 1
- length = 1
- 방향 = 0(동쪽)

차근차근 살펴봅시다

#알튜비튜

방향 변환 정보

3 D
15 L
17 D



front (1, 2) back

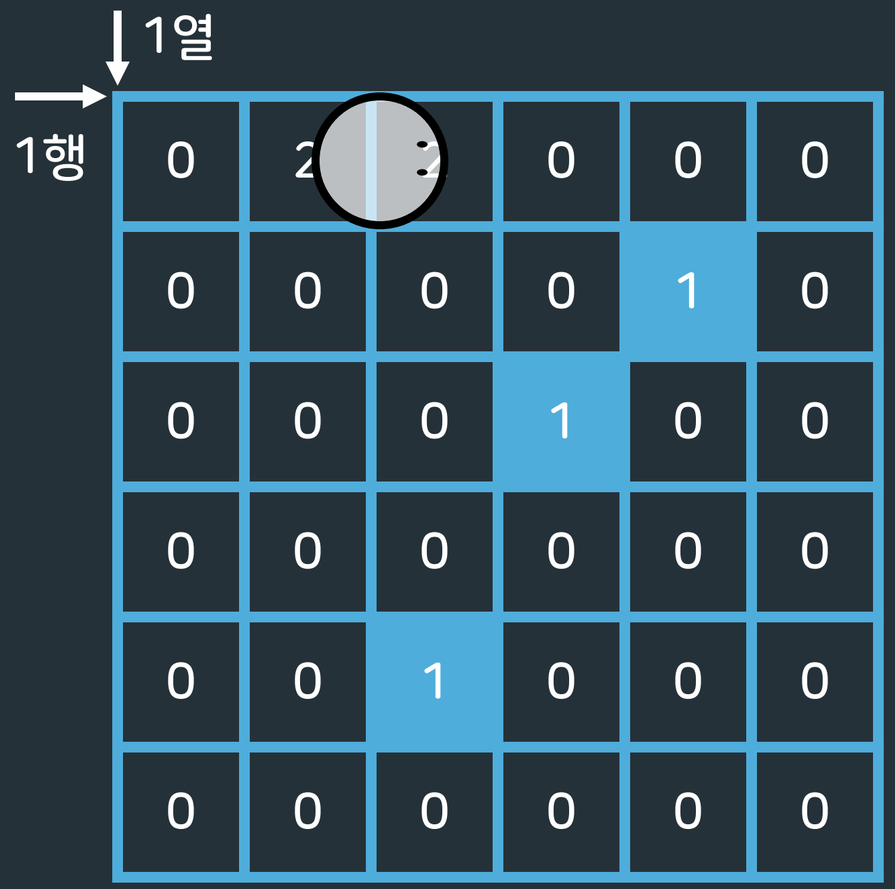
- time = 1
- length = 1
- 방향 = 0(동쪽)

차근차근 살펴봅시다

#알튜비튜

방향 변환 정보

3 D
15 L
17 D



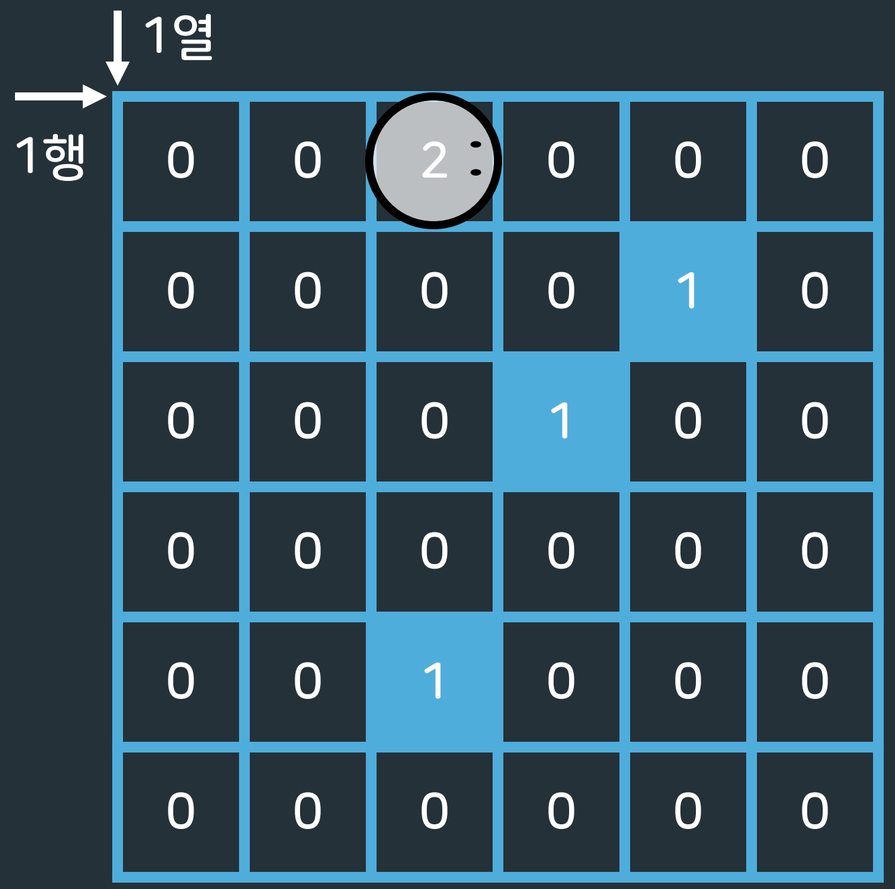
- time = 2
- length = 1
- 방향 = 0(동쪽)

차근차근 살펴봅시다

#알튜비튜

방향 변환 정보

3 D
15 L
17 D



front (1, 3) back

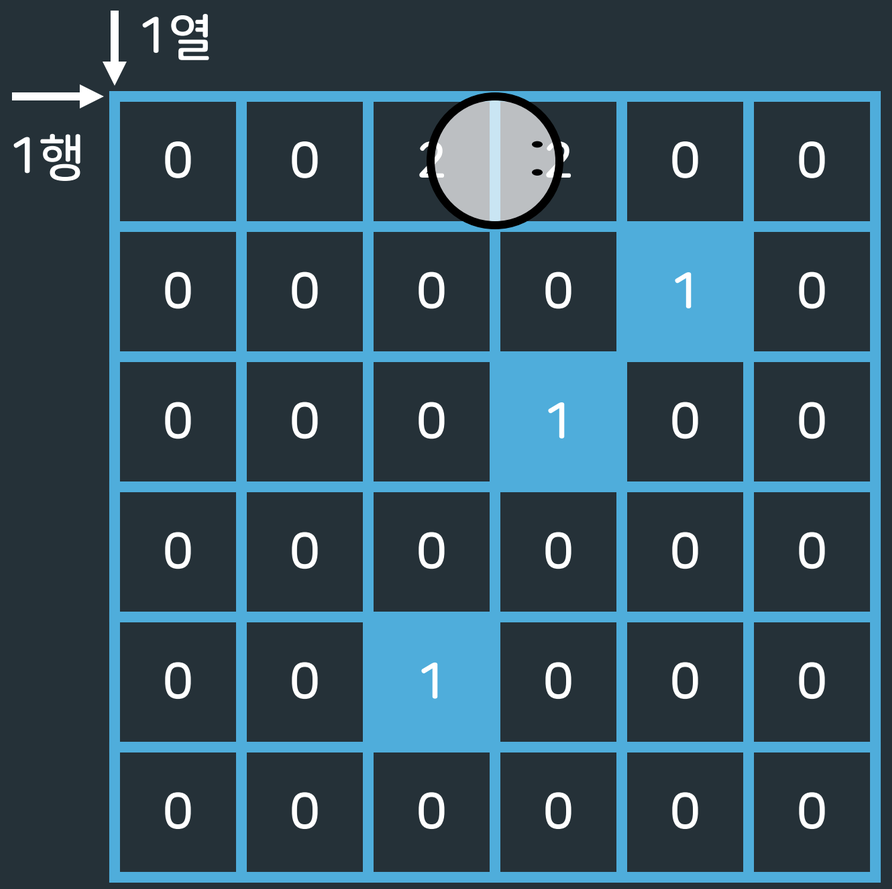
- time = 2
- length = 1
- 방향 = 0(동쪽)

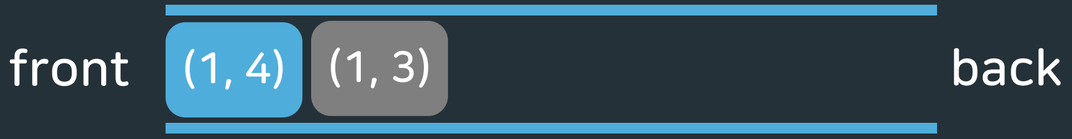
차근차근 살펴봅시다

#알튜비튜

방향 변환 정보

3 D
15 L
17 D





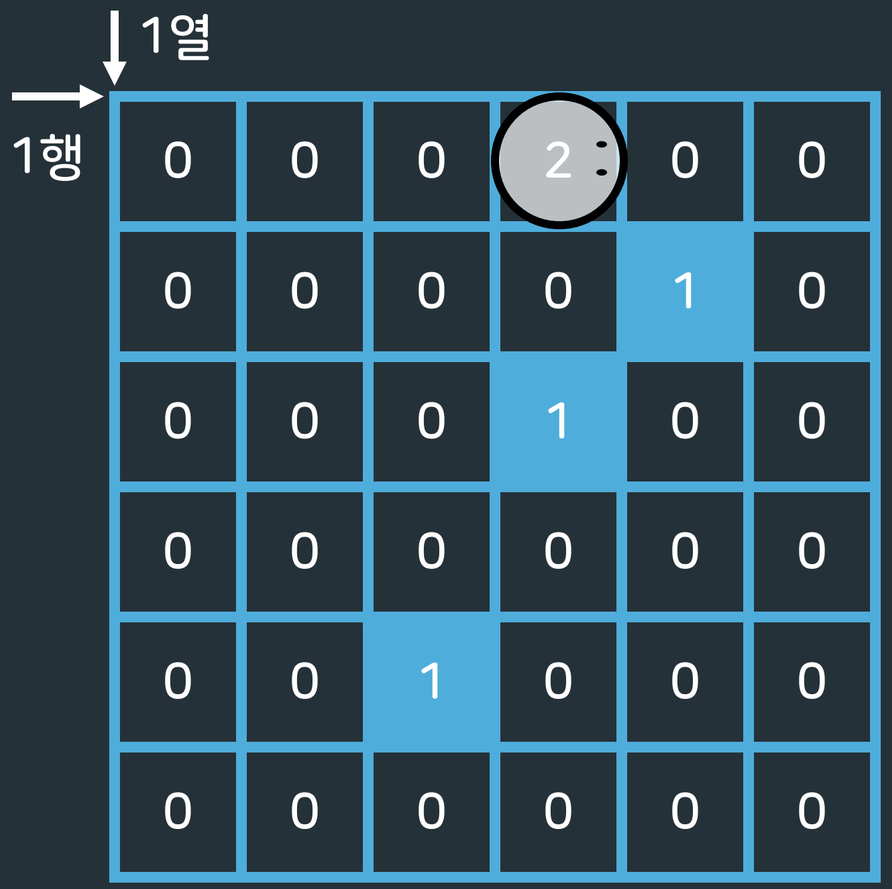
- time = 3
- length = 1
- 방향 = 0(동쪽)

차근차근 살펴봅시다

#알튜비튜

방향 변환 정보

3 D
15 L
17 D



front (1, 4) back

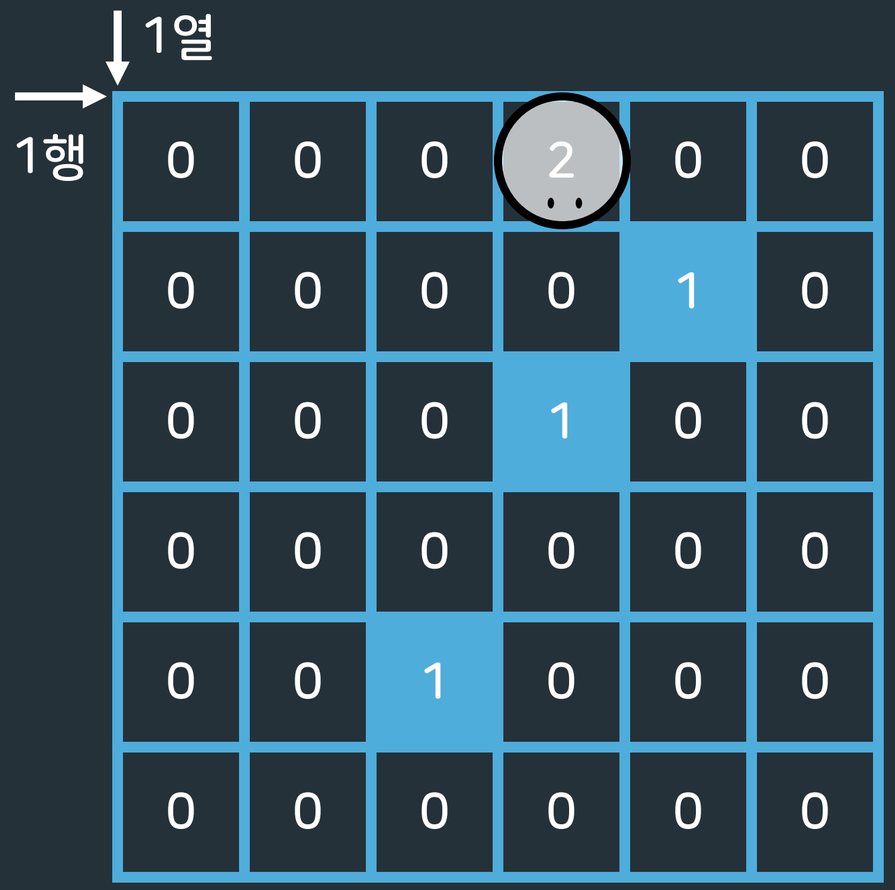
- time = 3
- length = 1
- 방향 = 0(동쪽)

차근차근 살펴봅시다

#알튜비튜

방향 변환 정보

3 D
15 L
17 D



front (1, 4) back

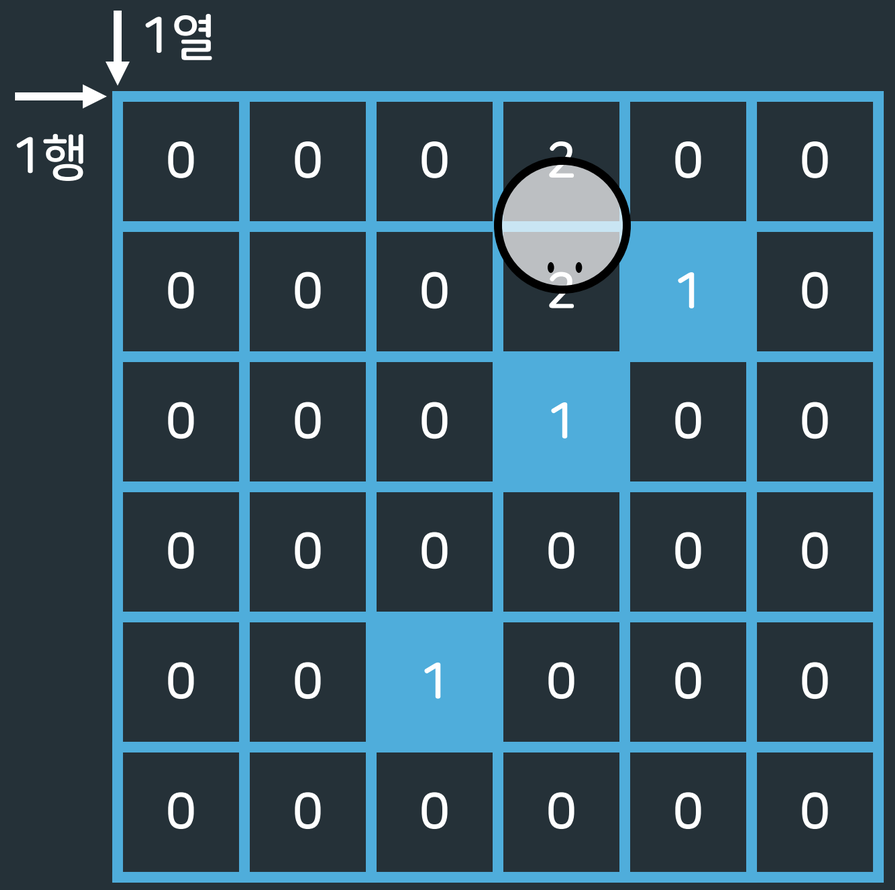
- time = 3
- length = 1
- 방향 = 1(남쪽)

차근차근 살펴봅시다

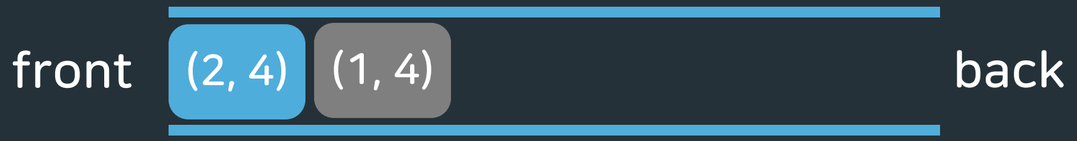
#알튜비튜

방향 변환 정보

~~3 D~~
15 L
17 D



- time = 4
- length = 1
- 방향 = 1(남쪽)

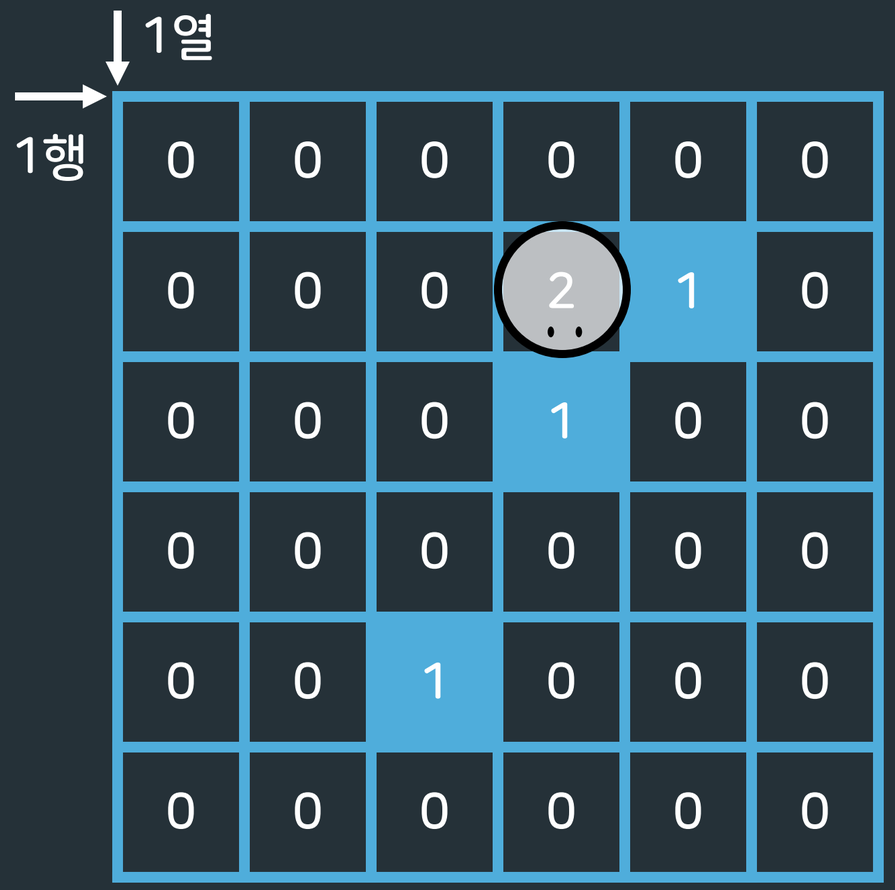


차근차근 살펴봅시다

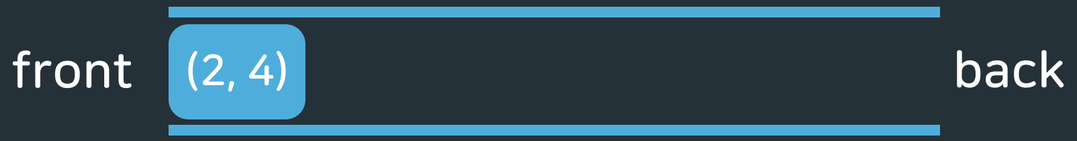
#알튜비튜

방향 변환 정보

~~3 D~~
15 L
17 D



- time = 4
- length = 1
- 방향 = 1(남쪽)

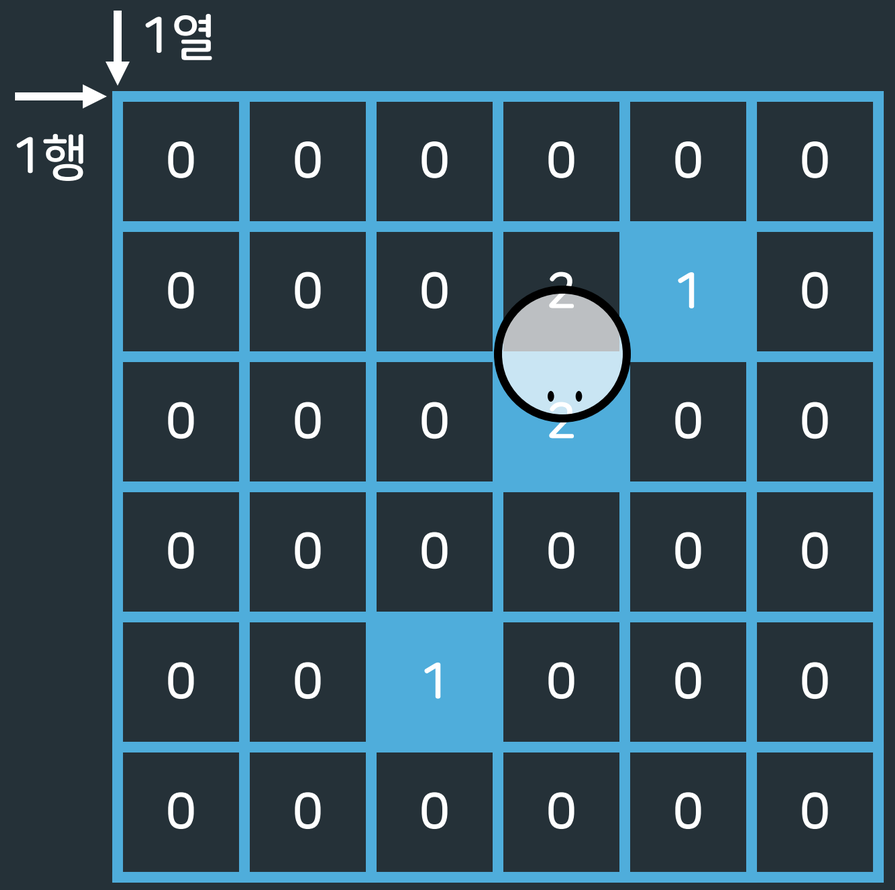


차근차근 살펴봅시다

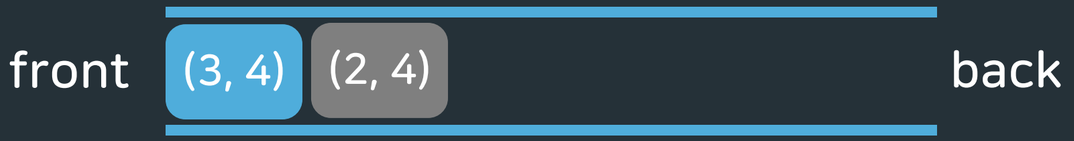
#알튜비튜

방향 변환 정보

~~3 D~~
15 L
17 D



- time = 5
- length = 1
- 방향 = 1(남쪽)

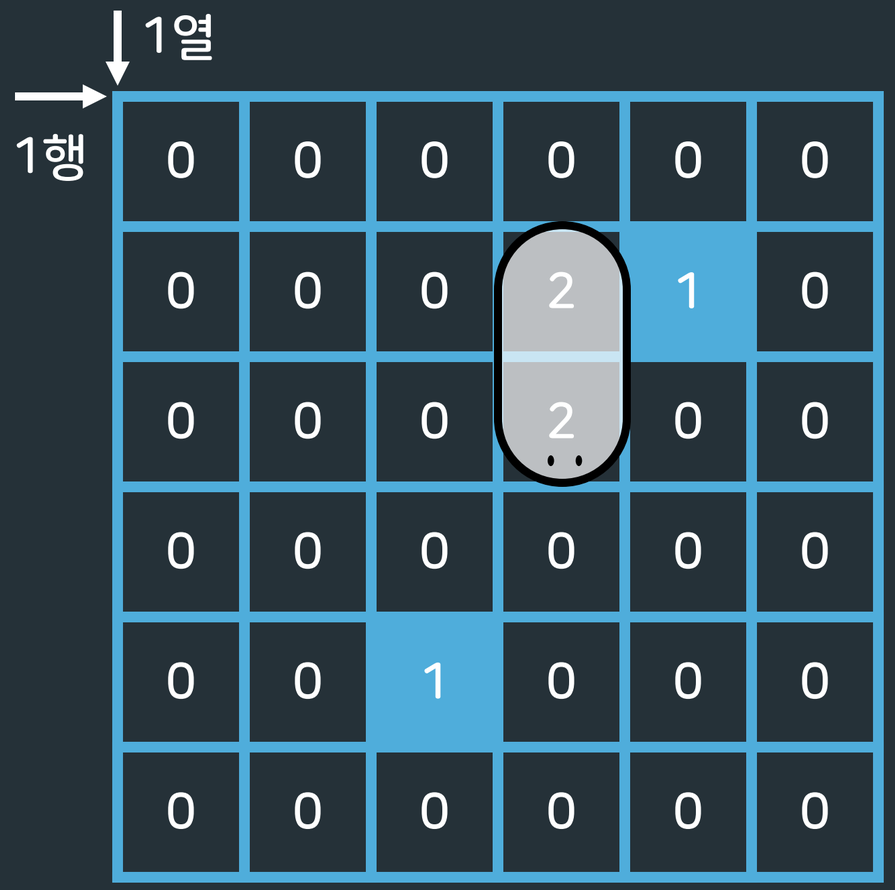


차근차근 살펴봅시다

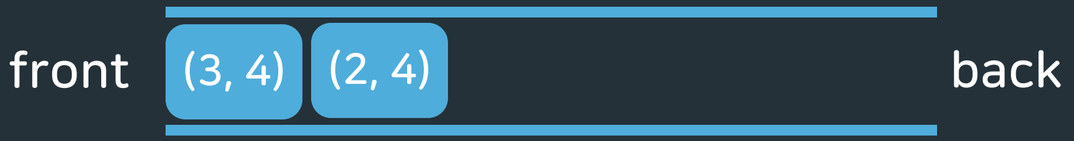
#알튜비튜

방향 변환 정보

~~3 D~~
15 L
17 D



- time = 5
- length = 2
- 방향 = 1(남쪽)

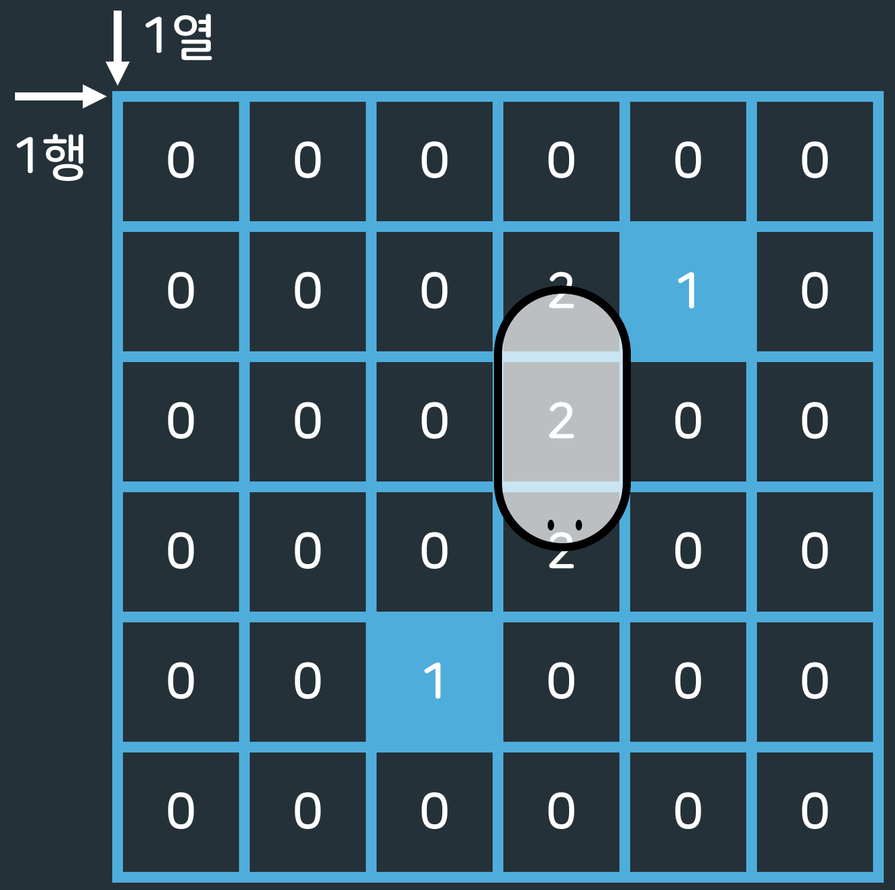


차근차근 살펴봅시다

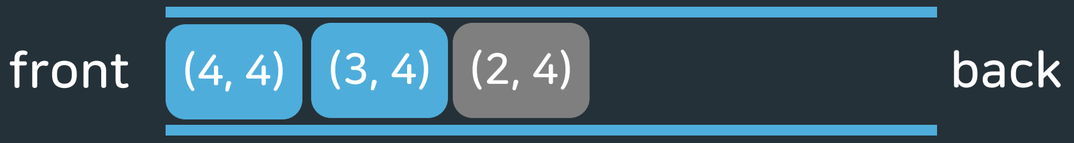
#알튜비튜

방향 변환 정보

~~3 D~~
15 L
17 D



- time = 6
- length = 2
- 방향 = 1(남쪽)

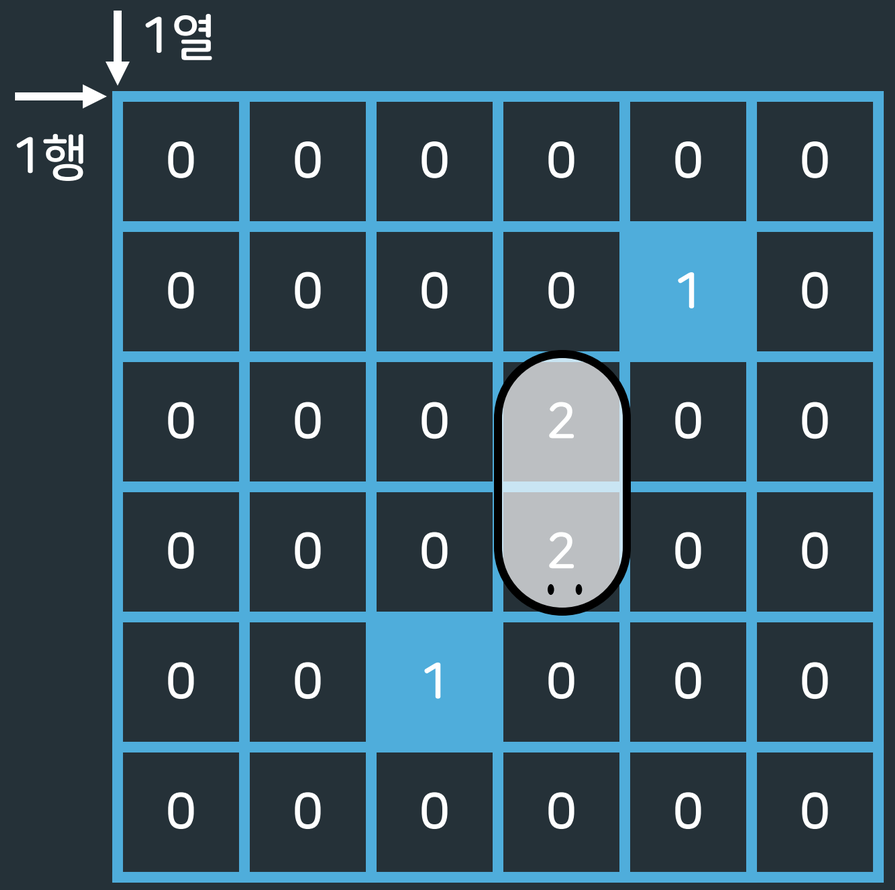


차근차근 살펴봅시다

#알튜비튜

방향 변환 정보

~~3 D~~
15 L
17 D



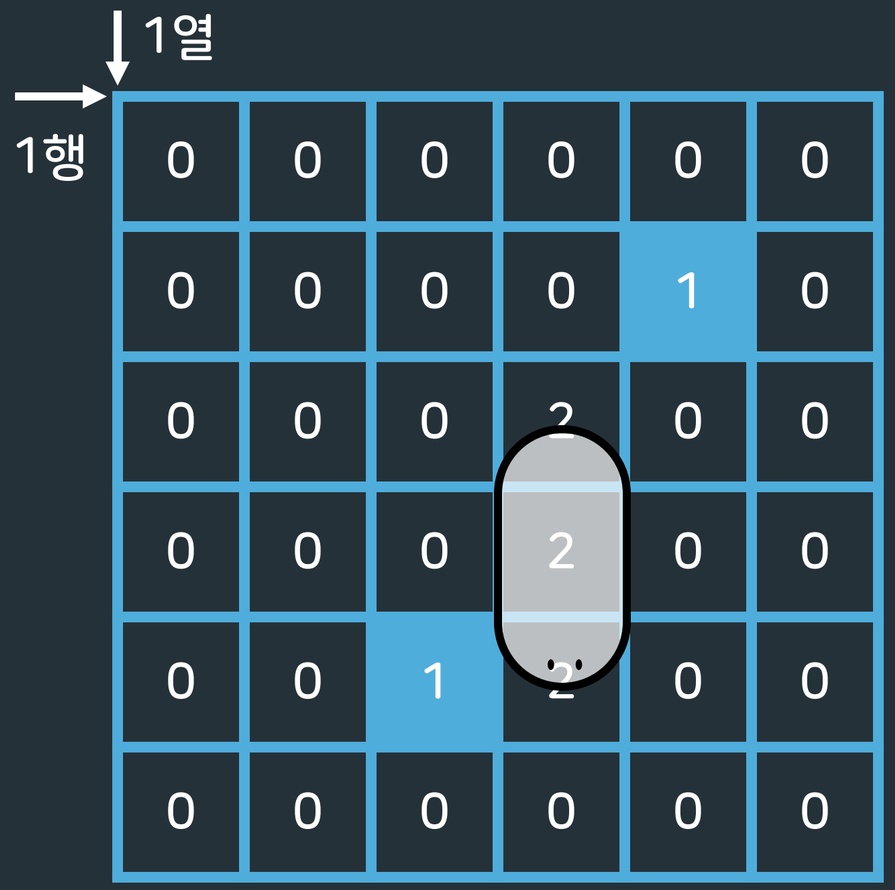
- time = 6
- length = 2
- 방향 = 1(남쪽)

차근차근 살펴봅시다

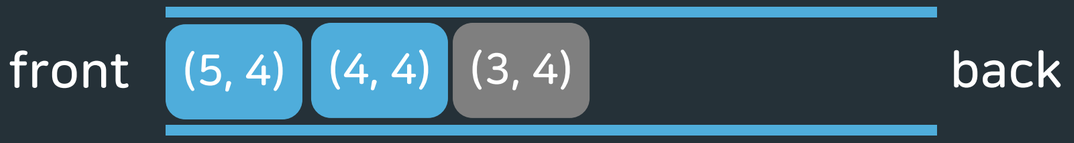
#알튜비튜

방향 변환 정보

~~3 D~~
15 L
17 D



- time = 7
- length = 2
- 방향 = 1(남쪽)

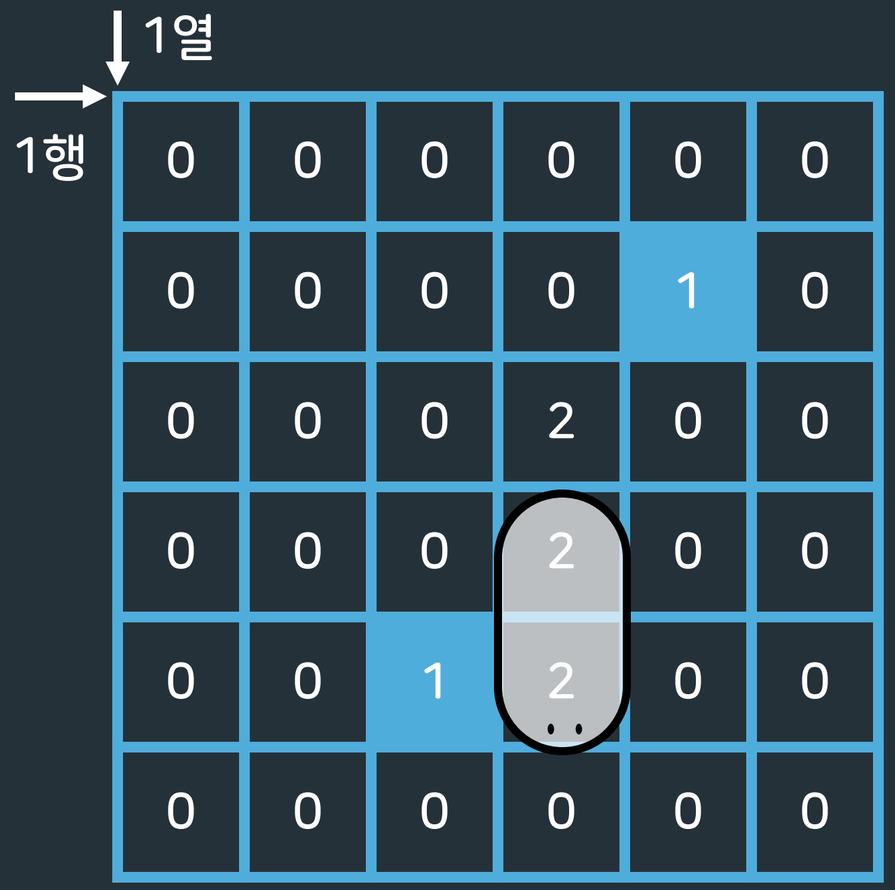


차근차근 살펴봅시다

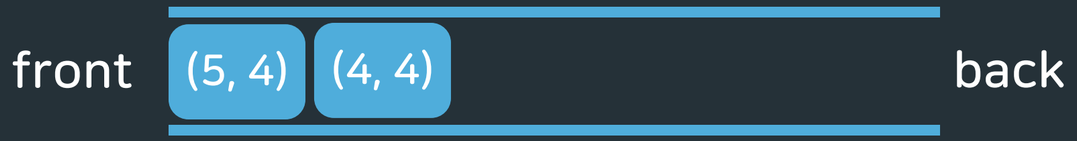
#알튜비튜

방향 변환 정보

~~3 D~~
15 L
17 D



- time = 7
- length = 2
- 방향 = 1(남쪽)

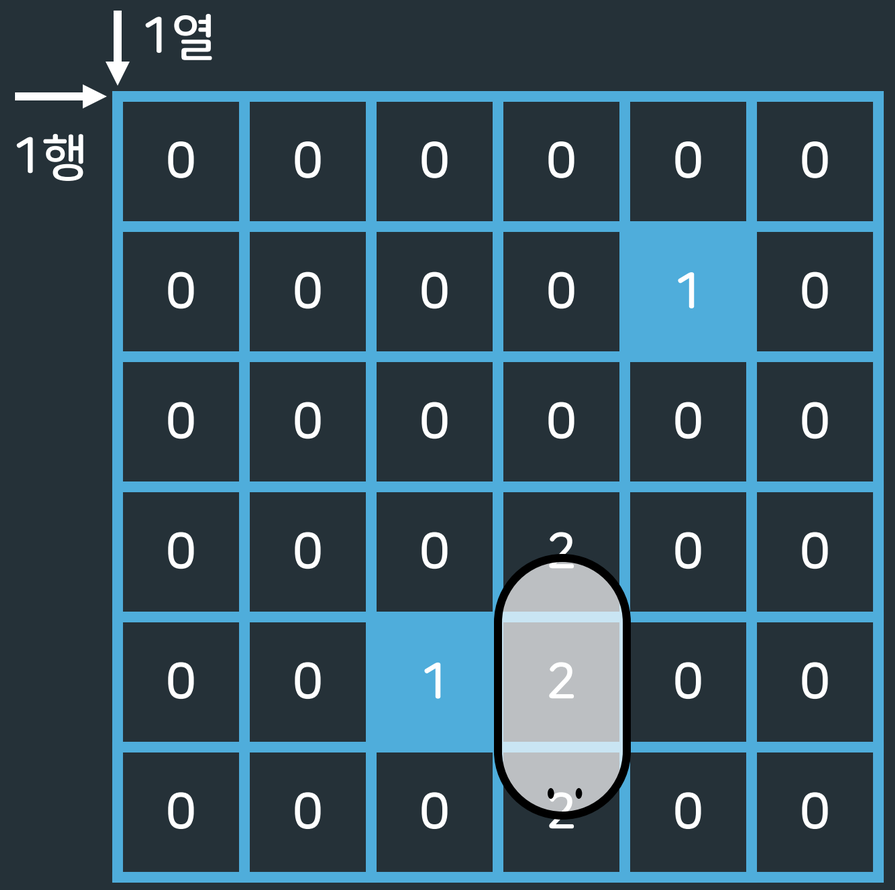


차근차근 살펴봅시다

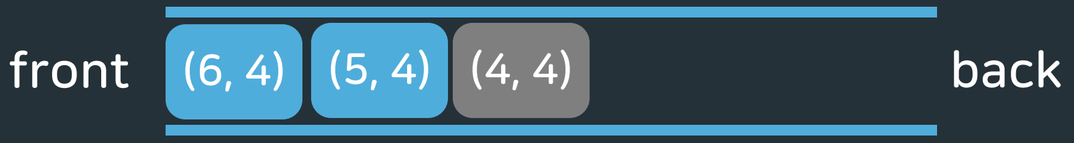
#알튜비튜

방향 변환 정보

~~3 D~~
15 L
17 D



- time = 8
- length = 2
- 방향 = 1(남쪽)

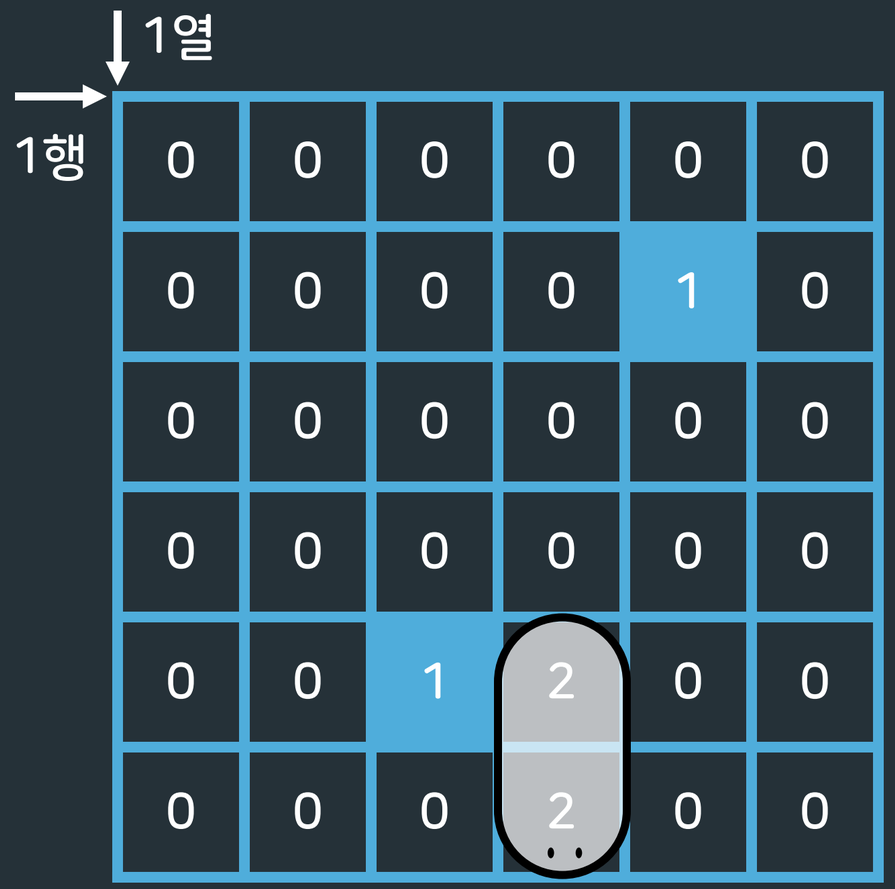


차근차근 살펴봅시다

#알튜비튜

방향 변환 정보

~~3 D~~
15 L
17 D



- time = 8
- length = 2
- 방향 = 1(남쪽)

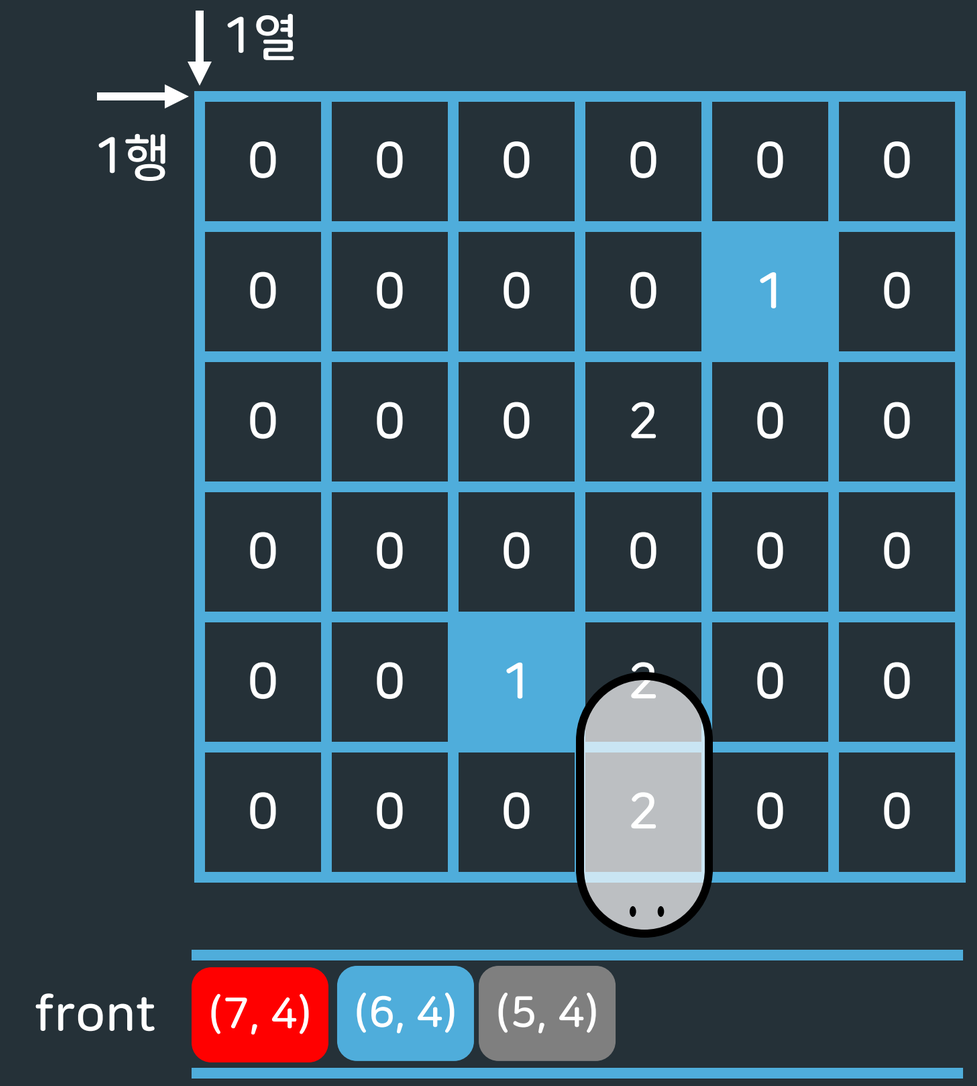
front (6, 4) (5, 4) back

차근차근 살펴봅시다

#알튜비튜

방향 변환 정보

~~3 D~~
15 L
17 D



- time = 9
- length = 2
- 방향 = 1(남쪽)

=> 게임 종료